

```

/*
Ant Project. Otago Polytechnic, New Zealand. 2009 3rd Year B.I.T. project for
Jun Cui, Gareth Dorset and Trevor Farquharson
-::~*_.*:-

Client: Otago Museum

Developers: Jun Cui - 3rd year student Otago Polytechnic
            Gareth Dorset - 3rd year student Otago Polytechnic
            Trevor Farquharson - 3rd year student Otago Polytechnic

Mentors: Patricia Haden - Otago Polytechnic
         Hamish Smith - Otago Polytechnic
         Sam Mann - Otago Polytechnic

*/

/**
 * @author >> Justin Windle
 * @link >> soulwire.co.uk
 * @version >> V1
 */

/* The code contained below is the original code by Justin Windle, full credit for making the motion tracker
work goes to him
any modifications to the code here will be values of variables required to fine tune for our
purposes.

We had full permission to use and modify this code.*/

package com.soulwire.media
{
    import com.soulwire.geom.ColourMatrix;
    import flash.display.BitmapData;
    import flash.display.BlendMode;
    import flash.filters.BlurFilter;
    import flash.filters.ColorMatrixFilter;
    import flash.geom.Matrix;
    import flash.geom.Point;
    import flash.geom.Rectangle;
    import flash.media.Video;

    public class MotionTracker extends Point
    {
        /*
        =====
        | Private Variables | Data Type
        =====
        */

        private static const DEFAULT_AREA: int = 10;
        private static const DEFAULT_BLUR: int = 20;
        private static const DEFAULT_BRIGHTNESS: int = 10;
        private static const DEFAULT_CONTRAST: int = 50;

        private var _src: Video;
        private var _now: BitmapData;
        private var _old: BitmapData;

        private var _blr: BlurFilter;
        private var _cmx: ColourMatrix;
        private var _col: ColorMatrixFilter;
        private var _box: Rectangle;
        private var _act: Boolean;
        private var _mtx: Matrix;
        private var _min: Number;

        /*
        =====
        | Constructor
        =====
        */

        /**
         * The MotionTracker class will track the movement within video data
         *
         * @param source A video object which will be used to track motion
         */

        public function MotionTracker( source:Video )
        {
            super();

            input = source;

            _cmx = new ColourMatrix();
            _blr = new BlurFilter();

            blur = DEFAULT_BLUR;
            minArea = DEFAULT_AREA;
            contrast = DEFAULT_CONTRAST;
            brightness = DEFAULT_BRIGHTNESS;
        }
    }
}

```

```

=====
| Public Methods
=====
*/

/**
 * Track movement within the source Video object.
 */

public function track():void
{
    _now.draw( _src, _mtx );
    _now.draw( _old, null, null, BlendMode.DIFFERENCE );

    _now.applyFilter( _now, _now.rect, new Point(), _col );
    _now.applyFilter( _now, _now.rect, new Point(), _blr );

    _now.threshold( _now, _now.rect, new Point(), '>', 0xFF333333, 0xFFFFFFFF );
    _old.draw( _src, _mtx );

    var area:Rectangle = _now.getColorBoundsRect( 0xFFFFFFFF, 0xFFFFFFFF, true );
    _act = ( area.width > ( _src.width / 100 ) * _min || area.height > ( _src.height / 100 ) *
_min );

    if ( _act )
    {
        _box = area;
        x = _box.x + ( _box.width / 2 );
        y = _box.y + ( _box.width / 2 );
    }
}

/*
=====
| Getters + Setters
=====
*/

/**
 * The image the MotionTracker is working from
 */
public function get trackingImage():BitmapData { return _now; }

/**
 * The area of the image the MotionTracker is working from
 */
public function get trackingArea():Rectangle { return new Rectangle( _src.x, _src.y,
_src.width, _src.height ); }

/**
 * Whether or not movement is currently being detected
 */

//public function movedetected(): Boolean {return _act;}
public function get hasMovement():Boolean { return _act; }

/**
 * The area in which movement is being detected
 */
public function get motionArea():Rectangle { return _box; }

/* INPUT */

/**
 * The video (usually created from a Camera) used to track motion
 */
public function get input():Video { return _src; }
public function set input( v:Video )
{
    _src = v;
    if ( _now != null ) { _now.dispose(); _old.dispose(); }
    _now = new BitmapData( v.width, v.height, false, 0 );
    _old = new BitmapData( v.width, v.height, false, 0 );
}

/* BLUR */

/**
 * the blur being applied to the input in order to improve accuracy
 */
public function get blur():int { return _blr.blurX; }
public function set blur( n:int ):void { _blr.blurX = _blr.blurY = n; }

/* BRIGHTNESS */

/**
 * The brightness filter being applied to the input
 */
public function get brightness():int { return _cmx.brightness; }
public function set brightness( n:int ):void
{
    _cmx.brightness = n;
    _col = new ColorMatrixFilter( _cmx.getMatrix() );
}

/* CONTRAST */

/**
 * The contrast filter being applied to the input

```

```

    */
    public function get contrast():int { return _cmx.contrast; }
    public function set contrast( n:int ):void
    {
        _cmx.contrast = n;
        _col = new ColorMatrixFilter( _cmx.getMatrix() );
    }

    /* MIN AREA */

    /**
     * The minimum area (percent of the input dimensions) of movement to be considered movement
     */
    public function get minArea():int { return _min; }
    public function set minArea( n:int ):void
    {
        if ( n < 0 ) return;
        _min = n;
    }

    /* FLIP INPUT */

    /**
     * Whether or not to flip the input for mirroring
     */
    public function get flipInput():Boolean { return _mtx.a < 1; }
    public function set flipInput( b:Boolean )
    {
        _mtx = new Matrix();
        if (b) { _mtx.translate( -_src.width, 0 ); _mtx.scale( -1, 1 ); }
    }
}

}

```